(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 762 789 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
12.03.1997 Bulletin 1997/11

(51) Int Cl.[6]: H04Q 7/24, H04Q 7/38

(21) Application number: 96305930.8

(22) Date of filing: 14.08.1996

(84) Designated Contracting States:
CH DE FR GB LI

(30) Priority: 22.08.1995 US 517938

(71) Applicant: AT&T IPM Corp.
Coral Gables, Florida 33134 (US)

(72) Inventors:
- La Porta, Thomas F.
Thornwood, New York 10594 (US)
- Veeraraghavan, Malathi
Atlantic Highlands, New Jersey 07716 (US)
- Ramjee,Ramachandran
Matawan, New Jersey 07733 (US)

(74) Representative:
Watts, Christopher Malcolm Kelway, Dr. et al
Lucent Technologies (UK) Ltd,
5 Mornington Road
Woodford Green Essex, IG8 0TU (GB)

(54) Based migrating user agents for personal communication services

(57) The present invention is a user process that resides in network nodes to act as an agent for the mobile terminals in PCS environments. The user process handles all negotiation and complex signaling functions for the user, thus reducing the amount of signaling traffic that must travel over the valuable air interface. To achieve low call establishment times the user process is migrated as users move. Three embodiments of mechanisms for migrating the user process are disclosed. The NAIVE approach enables the amount of data size to be transferred to be optimized, leading to low overhead. This approach also provides flexibility when migrating across heterogeneous environments. A second alternative, termed the TERN approach, is promising for cases when the program is compute intensive and asynchronous migration is essential. The last approach considered, Condor, provides high reliability in the form of checkpointing, but incurs a high migration delay and has high memory requirements for the network processors.

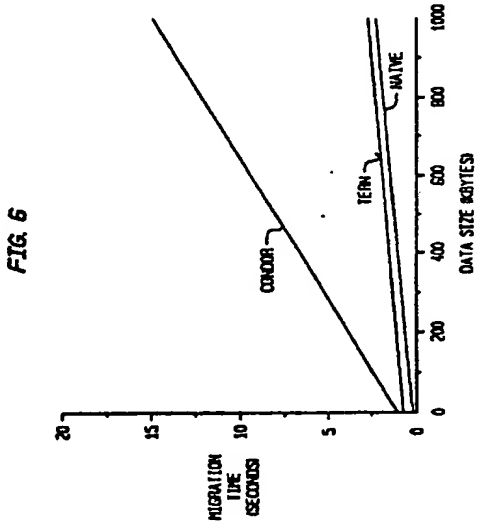


FIG. 6

EP 0 762 789 A2

to as clusters 12. Within each cluster 12 are User Signaling Servers 14, Call Servers 16 and Connection Servers 18. The User Signaling Servers 14 house multiple user processes 20 on behalf of mobile terminals operating within the cluster. Within a cluster 12, a mobile station 24, also referred to herein as a mobile terminal, is assigned to a User Signaling Server 14. The user process 20 communicates with the mobile terminal through a base station 23 currently serving the mobile terminal, wherein the base station includes, for example, a channel server and radio port 28 to wirelessly communicate with the mobile. The Call Server 16 maintains the network state of the call and the Connection Servers 18 are responsible for establishing the communication path over which user information (voice or data) is transmitted. Coupled between the connection servers 18 and the base station 23 are a plurality of channel servers 25 and switches 17, for example, ATM switches, each of which is also involved in establishing the communication path as would be understood by one skilled in the art.

Each network contains multiple Location Servers 22 which are responsible for tracking the location of a mobile terminal within a network. The Location Servers 22 are not associated with a cluster, but track the movement of mobile terminals across cluster boundaries. When a mobile terminal is powered on it registers with a Location Server. For a more detailed description of a cluster-based network, see related U.S. Patent Application Serial No. 08/324,427, entitled Method And System For Distributed Control In Wireless Cellular And Personal Communication Systems, having a filing date of October 17, 1994, and being incorporated herein by reference.

Referring to FIG. 2, the registration procedure of a mobile terminal 24 is illustrated. This registration includes a profile 26 provided by the mobile terminal. This profile 26 contains compatibility information of the mobile terminal 24, which applications are running, and any other user specific information that is required to create the user process. One example of User Profile data is the MOB_TERM parameter of IS-95 which indicates whether the cellular user is willing to accept incoming calls. The Location Server 22 is responsible for recording the current cluster of the mobile terminal 24, and instructing a User Signaling Server 14 to instantiate a user process on behalf of the mobile terminal. This user process is loaded with the user profile so that it can act on behalf of the user.

An analogous flow for power down registration results in the location server instructing the User Signaling Server to terminate the execution of the user process.

To illustrate the functionality of the user process, the scenarios for call setup are next described. When a call is placed to a mobile terminal 24, as shown in Figure 3, the network offers the call to the user process 20 representing the called mobile terminal. If there is any negotiation or compatibility checking that must take place between the called mobile terminal 24 and the calling party or the network, it occurs at this point of the call establishment through further interaction with the user process. If the user process 20 decides to accept the call, it pages its mobile terminal 24 to determine its exact location, for example, a specific radio-port 28 within a cluster. Thus the mobile is tracked to the granularity of a cluster when it is idle and its exact location determined at call setup time, thereby optimizing on the trade-off between the number of registrations and the amount of paging required.

When the mobile terminal 24 wishes to place a call, as shown in Figure 4, it signals to its user process 20 and requests a call of a certain type. The user process 20 maps the call type into specific services and applications that are executing on the mobile terminal, and generates a call request. If there is any negotiation that must take place between the mobile terminal and the called party, it is done through interaction with the user process 20.

As has been briefly described, the present invention addresses the challenges of user process mobility management by enabling the user process to essentially migrate along with the mobile user. When a mobile terminal 24 crosses cluster boundaries, it is assigned to a new User Signaling Server 15 by the Location Server 22. According to the present invention, the user process migrates from its current location to the new User Signaling Server 15. The process is required to migrate, and not simply be re-instantiated, because it contains the current state of the user which must be preserved. These parameters include, for example, the identifiers of any calls in which the mobile terminal is currently active. In addition, we do not wish to repeatedly download the user profile information over the air interface. Besides creating unnecessary load on the air interface, this could pose security risks for the user.

Because a migration between clusters is a managed event, we do not require transparent process migration which has been the aim of many previous process migration efforts. In the instant case, the process itself requires that it be executing at a new location. Therefore, the process is aware it is migrating, can plan for the migration, and assist in the migration. Because a specific process is migrating and not an arbitrary one, use can be made of information specific to the process being migrated. For example, the user process which has to migrate does not have any open file descriptors and does not use any signals. Thus, the present invention mechanisms need not have special purpose signal handling or file handling routines.

Migration, in the instant case, however, is complicated by the fact that communication logic is strongly coupled with the processing logic of the application and thus, means must be provided for re-establishing the communication channels of the process after it has migrated. In "The DIANA approach to mobile computing," T. Ahmad, M. Clary, O. Densmore, S. Gadol, A. Keller and R. Pang, Mobidatajournal, Vol. 1, No. 1, Nov. 1994, a new application architecture is proposed which decouples the communication logic from the processing logic of the application. This feature is particularly appealing as the migration mechanism will have to deal only with migrating the processing logic-specific

components, not the communication logic.

The various steps involved in th migration process are illustrated in Figure 5. The User Signaling Servers (old 14 and target 15) reside on two differ nt nodes in different clusters. When a mobile station 24 changes clust rs, it gen rates a REGISTRATION message to its Location Server 22. The Location Server 22 signals to the targ t User Signaling Server 15 to migrate the process (MIGRATE). The target User Signaling Server communicates with the old User Signaling Server 14 which transfers all relevant information about the user process 20 to the target User Signaling Server 15, for example, all data and stack pages, the state of the user, and all user profile information. The old User Signaling Server 14 also maintains a stub process in place of the user process 20 to buffer any messages arriving for the user process while migration is underway. The new user process 21 requests any buffered messages from the stub process residing on the old User Signaling Server 14 and resumes its normal processing. The stub process exits after an idle time-out period.

Two major benefits are derived by maintaining and migrating a user process in the network: less air signaling traffic and lower call establishment times. The benefit of reducing the amount of signaling traffic generated over the shared air signaling channel results from maintaining a user process in the network. As seen in Figures 3 and 4, few messages are required between the mobile terminal 24 and the user process 20 to establish a call. Instead, the bulk of the signaling message exchange occurs between the user process and other users or servers in the network. In addition, if the user process does not wish to accept a call, no paging of the user terminal is required. This reduces the amount of paging traffic required in the network, which is a known bottleneck. The benefit of lowering call establishment times results from the fact that the user process is located near the mobile terminal.

Another approach is to anchor the user process in the user signaling server in which it was created during power-up registration. As the user moves, radio ports can be updated with the location of the anchored user process. This allows the new radio port of the mobile terminal to send call requests or other messages generated from the mobile to the appropriate user process. Similarly, when the mobile changes clusters that are used for mobile tracking, the user process needs to be updated with the new paging area data allowing it to page the mobile for incoming call delivery. The disadvantage is that since the mobile terminal may be far removed from the user signaling server where its user agent (UP) is located, signaling message transit times and hence call establishment times may increase.

Figure 9 illustrates two schemes that can be used to allow a radio port 28 to communicate with any user signaling server 14. The first scheme is via connectionless datagram routers 13. In this case, when mobile terminal 24 moves from a first radio port to a new radio port 11, the new radio port, and possibly other radio ports within the cluster are updated with a datagram address of the new user signaling server.

In the second scheme, connection-oriented signaling links, e.g., Asynchronous Transfer Mode virtual channel connection 15, 17 may be used from radio ports to user signaling servers. These signaling links can be pre-established or established as the users move. The advantage to the connectionless approach is that radio port updates are simpler than connection setup procedures. On the other hand, connection oriented signaling is typically faster than signaling through datagram routers.

In any event, when analyzing migration of the user process, the associated overhead costs were analyzed to provide justification for possible implementation of the migrating user process. When considering signaling load, we are most concerned about signaling messages that must travel outside of a localized area which may congest long distance, or cross-network, signaling links.

The proposal to migrate the user process as a user changes clusters indicates that the only overhead incurred by migrating the user process is the migration procedure itself. As can be seen in Figures 2-4, there is no extra signaling overhead associated with registration procedures or call control resulting from the fact that the user process migrates.

The flow for changing clusters as shown in Figure 5 indicates that the MIGRATE and MIGRATE_DONE commands may travel on long distance signaling links as the location servers are located independently from the cluster in which the mobile users are located. The TRANSFER command in Figure 5 requires a message exchange between two neighboring clusters. Therefore, the long distance signaling overhead associated with migrating the user process is limited to the MIGRATE/ MIGRATE_DONE message exchange between the user signaling servers and the location servers. These messages total to approximately 150 bytes.

To estimate the overhead incurred in migration, we use common assumptions for characteristics of a PCS network, summarized in Table 1. Assuming that the direction of the movement of the terminals is uniformly

Table 1:

Model Parameters	
Parameter	**Value**
Total number of mobile terminals	2.87 million
Average call origination/arrival rate	1.4/hour/terminal

Table 1: (continued)

Mod l Param ter	
Parameter	Value
Mean Density of Terminals, ρ	390/sq. km
Average speed of a mobile terminal, v	5.6 km/hr
Cluster (square) edge length, L	30.3 km
Clusters per network	128

distributed over [0, 2π], and using a fluid flow mobility model, for example, "Influence of mobile station on the performance of a Radio Mobile cellular network," R.Thomas, H. Gilbert and G.Mazziotto, Proc. of 3rd Nordic Sem., paper 9.4, Copenhagen, Denmark, Sept. 1988, the rate of mobiles crossing a cluster boundary is

$$R=\rho vL/\pi$$

Given the values of Table 1, a single cluster experiences 5.85 boundary crossings per second. Accounting for 128 clusters in the network, our proposal incurs 900 Kbps long distance signaling overhead network-wide. This translates to about 7 Kbps at each cluster. Thus, it is believed that the overhead incurred in migrating the user process to enable faster connection setup is a worthwhile trade-off.

There are several considerations when choosing schemes for process migration. First, it is required that the delay in migrating a process from one node to another be as small as possible. This is because the user process is not processing any messages while it is migrating, and thus the time taken to migrate will manifest in the call setup delay if messages happen to get queued at the old location during migration. On the other hand, delay in initiation of migration is not a major concern because the user process remains active while waiting for the migration to be initiated. Flexibility in operating environment is another concern, and thus rules out some schemes which require a special purpose operating system.

With these requirements in mind, three different embodiments are considered for implementation of the present invention, each of which offer the required flexibility. A performance comparison is then made in terms of the speed of migration for different loads (which in this case is the data size of the migrant process). In each case, it is assumed that all nodes have the executable code for the user process resident, as this will reduce the amount of information that must be transferred between the source and destination nodes, and will in turn reduce the delay in migration.

The approaches considered, the Condor approach, the TERN approach, and the NAIVE approach, are flexible in terms of operating environment as they do not require access to the internals of the operating system. The Condor approach can be run entirely at the user level. The only requirement for the TERN approach is that it must be possible to access the kernel virtual memory of the system. The NAIVE approach requires the process to manage the migration itself by saving its variables and re-reading them upon migration.

Most process migration schemes do not handle system calls relating to communication, for example the socket system call, since there is the problem of hidden state in the form of transit messages. In the instant case, we only require that all transit messages be delivered in sequence to the process after its migration, and are not concerned as to whether the message intended to the user process in machine A was actually delivered to the user process after it had migrated to machine B. The problem of redirection of messages is similar to the one faced by the mobile IP mechanisms, in which one needs to ensure that packets reach the mobile station at its new address as the mobile station moves. In our case, we create a temporary stub which stores any arriving messages during the migration phase, and forwards them to the migrated process after it has stabilized.

The first preferred embodiment of the user process migration scheme is based on the Condor migration system, M.Litzkow and M.Solomon, "Supporting checkpointing and process migration outside the Unix Kernel," in Usenix Winter Conference, San Francisco, California, 1992 and M.J.Litzkow, M.Livny and M.W.Mutka, "Condor - A Hunter of Idle Workstations," 8th Inter- national Conference on Distributed Computing Systems, Jan Jose, Calif., June 1988. The Condor system supports process migration entirely at the user level and has been implemented on UNIX platforms. At the core of the technique is the notion of checkpointing; the process is checkpointed at one machine and resumed at another machine. As far as the process is concerned, the migration appears to be the arrival of a signal and a *setjmp* call, which occurs at the original machine, and a return from a *longjmp* call which occurs at the new machine.

The basic idea behind the Condor approach is to dump the core of memory of an initial user process and then recreate the process at the n w destination. In other words, ev rything in memory is transferred and then the user process

is started. Referring to FIG. 7, there is shown an exemplary embodiment of a user process 20 according to the present invention. As shown, the user process 20 includes two main portions, a core 40 and executable 42. The core 40 includes a user area 44, stack 46 and data storage 48. The executable 42 includes text 50, initialized data 52 and symbol and debugging information 54. As mentioned previously, the Condor approach essentially checkpoints the user process from one machine to another. The new user process 21 which is checkpointed at the new machine (or user signaling server) includes the stack 46, symbol and debugging information 54, initialized data 52 and text 50.

The steps involved in implementation of the Condor approach to migrate the user process after the mobile station changes clusters are summarized in the flow diagram of FIG. 7A and are as follows:

1. The User Signaling Server (USS) sends a SIGTSTP signal to the user process. (110)
2. The user process forks off a child stub process that buffers the messages sent to the original user process. (120)
3. The parent user process now executes a *setjmp* to save important registers (PC. SP. etc.) in a buffer and dumps a core, thus exiting. (130)
4. The USS transfers information (data and stack pages) from the core to the target USS. (140)
5. The target USS creates the new user process using the text segment from the executable present in its node, and the stack and data segments received from the old USS. (140)
6. The user process establishes its communication channels.
7. The user process sends a READY message to the stub process and receives any outstanding messages that have been buffered. (150)
8. The user process executes a *longjmp* to continue execution from where it left off. (150)

Note that all these steps can be achieved at the user level. The new user process created is a checkpoint file and can be reviewed later in the case of a machine or network crash. Since Condor is a general purpose mechanism, migration can be initiated asynchronously, that is, the process can be stopped at any point in the code and migrated, thus ensuring a quick response to a migration call.

A second preferred embodiment of the present invention migration scheme is based on TERN, a transparent process migration mechanism described in M. Kannan, "TERN - migration mechanism and the communication layer, " B.Tech project report, Dept. of Computer Science, IIT Madras, 1992 and K.G.Venkatasubramanian, "Preprocessing and Run-Time System Call support for TERN," B.Tech project report, Dept. of Computer Science, IIT Madras, 1992. As opposed to Condor, a checkpoint file is not created in this scheme. The mechanism requires information, such as process data and stack sizes, of an executing process. This requires access to kernel virtual memory, which is usually restricted to the general user. This access requirement is not a concern since the User Signaling Servers are resident in a network node, and are thus in a curtailed service environment.

Referring to FIG. 8, an exemplary implementation of the migration approach is shown. As can be seen, a user process 20 is contained at a source node 70 and a destination node 72. The address space of each user process includes a text segment 74, a data segment 76 and a stack segment 78. Under TERN, the stack and data pages 76, 78 are directly copied from the virtual address space of the user process. The basic idea behind TERN is to start up a process at a destination and then copy stack and data pages onto the appropriate virtual address space. Thus a core dump is not required, resulting in greater efficiency as will be seen. TERN, like Condor, is a general purpose mechanism, and thus migration can be initiated asynchronously resulting in a quick response to a migration call.

The steps involved in migrating a user process using TERN are summarized in the flow diagram of FIG. 8A and are as follows:

1. The User Signaling Server (USS) on the source node stops the user process and transfers the arguments and the environment of the process to the target USS. (210)
2. The USS sets up a stub process to buffer any incoming messages. (210)
3. The USS on the target node *exec's* a new user process with the arguments and environment obtained from the source USS. (220)
4. The USS on the source node calculates the data and stack sizes of the migrant process and transfers this information to the target USS which ensures that the new user process expands its stack and data pages suitably. (220)
5. The data pages are transferred and written onto the virtual address space of the user process executing on the target User Signaling Server. (230)
6. The stack pages are transferred and written. (230)
7. The user process establishes its communication channels.
8. The user process sends a READY message to the stub process and receives any outstanding messages that have been buffered. (240)
9. The register contents are transferred and the new user process restarts from where it was interrupted. (250)

As mentioned earlier, one main difference between th present requirement f r process migration and other process migration mechanism requirem nts is that it is known, a priori, which process is going t migrate. Thus, we need not have a general purpose migration mechanism and can resort to transferring only the necessary logical variabl s in order to increase efficiency. This type approach is presented as the third preferred embodiment for the present invention user process migration scheme, wherein it is termed the NAIVE approach. The steps involved are summarized in FIG. 10 and are as follows:

1. The User Signaling Server on the target node *exec's* a new user process. (310)
2. The old user process establishes connection with the new user process. (320)
3. The old user process transfers the data structure containing all the variables pertinent to the migration. (330)
4. The new user process establishes its communication channels. (340)
5. The old user process becomes a stub process and forwards any messages to the new user process. (350)
6. The new user process resumes its processing. (360)

Here, the variables crucial to the migration are placed in a suitable data structure so that modifying the user process code will have minimum ramifications to the migration mechanism. Examples of pertinent variables include those parameter having to do with call connection and the state of the mobile. Relevant data pertaining to call connection may include, for example, type of connection, number of users, identification of the users, call type, call forwarding and whether or not the mobile station will accept incoming calls. Information pertaining to the state of the mobile may include, for example, terminal identities, what applications are running, e.g., Xserver running, compatibility checking, e.g., video card, etc.

Note that there are two major differences between the NAIVE approach and the TERN and Condor approaches. In the NAIVE approach, the migration procedures are closely tied to the program logic, therefore in the NAIVE approach the user process can be selective of what information will be transferred, which significantly increases the efficiency of the migration procedure. For example, if a mobile station is idle, then much less information need be transferred than if the mobile is actively engaged. In this instance the user process makes decisions as part of the program logic and adjusts the volume of data that is necessary to be transferred. In the TERN and Condor approaches, on the other hand, the migration procedures are independent of the program logic. Also, in the NAIVE approach, the migration may only be initiated at certain points in the code, i.e., when all of the crucial variables have been stabilized and saved, whereas in the TERN and Condor approaches the migration process may begin asynchronously.

Table 2 presents a summary of the three process migration mechanisms with relation to characteristics

Table 2:

Process Migration Mechanism Summary				
Method	**Independent of program logic**	**Works across heterogeneous platforms**	**Asynchronous initiation**	**Optimizes data to be migrated**
NAIVE	N	Y	N	Y
TERN	Y	N	Y	N
CONDOR	Y	N	Y	N

that are generally considered to be attractive in process migration mechanisms. The NAIVE approach has two possible disadvantages. First, it is not independent of program logic. It was found, however, that this did not overly complicate our implementation. In addition, it is this dependence that allows us to optimize the amount of data that must be transferred during the migration. The second possible disadvantage is that the process migration may not start asynchronously. As stated previously, this does not concern our application because the user process is active while waiting for migration to commence; we are more concerned with the time for migration to conclude once it has begun.

The TERN and Condor approaches have two possible disadvantages. First, the amount of data that must be transferred cannot be optimized. Also, neither approach will work across heterogeneous systems. The Condor approach has one additional disadvantage. It requires a large amount of memory on the User Signaling Server to accommodate the core dump during process migration.

The per-byte cost of these schemes and the performance related issues of the migration itself can now be examined. As discussed earlier, one of the most important requirements is to incur a low delay in migrating a process. Thus, the time taken to migrate for each of these three schemes was measured. Clearly, this time depends on the amount f information being transf rred between the User Signaling Servers.

The three schemes discussed were implemented on UNIX platforms. A typical data point was obtained by migrating a process between two machines located on different ethernet local area networks, and averaging the time taken to migrate the process. The data size was varied from 1Kbyte to 1Mbyte, and the time to migrate for each of the three approaches was measured. All three schemes varied linearly with data size as expected. However, the per-byte overhead differed in each of the approaches and was critical to the overall system performance.

Referring to FIG. 6, it can clearly be seen that the TERN and NAIVE approaches have low per-byte cost and scale well with increase in data size. The TERN approach performs better than Condor because it is much cheaper to write directly onto the virtual address space of a process rather than read/write onto the executable file.

A key to the performance of the actual system is the data size of the user process and how much of it is crucial to the migration. Current implementation of the user process has a data size of about 72 Kbytes which must be migrated when using the TERN approach. This takes, on average, 750 milliseconds to migrate on the UNIX platform. In the case of the NAIVE approach, we were able to optimize the size of the data to be migrated to between 45 and 100 bytes depending on the size of the profile and whether the user was active in a call or not. This takes, on average, 150 milliseconds to migrate. This illustrates the advantage of being able to optimize the amount of data to be transferred when using the NAIVE approach.

This difference is due to the fact that the user process consists of several protocol layers and libraries. When using the NAIVE approach, the user process migrates only when it is in a "stable" state; that is the state of the lower layer data elements are inconsequential as far as migration is concerned. While this may cause a delay in the initiation of migration, as previously stated, this does not concern our application.

The TERN approach is an attractive alternative if the user process evolves into a more processing intensive entity than in the present system. For example, in systems in which a user agent processes data being exchanged with the end device, it may not be possible to cleanly separate a small set of variables to be migrated. The Condor approach, on the other hand, is attractive only when checkpointing for reliability or other reasons is required.

Three embodiments of possible mechanisms for migrating the user process have been presented. The NAIVE approach appears to be the most promising; in that the amount of data size to be transferred can be optimized, leading to low overhead. This approach also provides flexibility when migrating across heterogeneous environments. A second alternative, termed the TERN approach, is promising for cases when the program is compute intensive and asynchronous migration is essential. The last approach considered, Condor, provides high reliability in the form of checkpointing, but incurs a high migration delay and has high memory requirements for the network processors.

From the above, it should be understood that the embodiments described, in regard to the drawings, are merely exemplary and that a person skilled in the art may make variations and modifications to the shown embodiments without departing from the spirit and scope of the invention. All such variations and modifications are intended to be included within the scope of the invention as defined in the appended claims.

Claims

1. A method of call processing in a communications network said network including mobile stations adapted for wireless communication, said method comprising the steps of:

   registering a mobile station with said network, wherein an individual user agent is established in response to registration of said mobile station at a signaling server within said network, and wherein said user agent is associated with said mobile station and includes an operating profile thereof: and
   utilizing said user agent in conjunction with said operating profile to handle call processing functions at said signaling server on behalf of said mobile station to thereby reduce signaling loads over an air interface of said communications network.

2. The method of Claim I, including the step of migrating said user agent from a first user signaling server in a first region of said network to a second user signaling server in a second region of said network in response to a move by said mobile station from said first region to said second region.

3. The method of Claim 1, wherein each mobile station and user signaling server are associated with a cluster, further including the step of migrating said user agent to a target user signaling server in a different target cluster when said mobile station moves to said target cluster, wherein operating parameters of said user agent are preserved for use at a new location proximate said mobile station, thereby reducing call establishment time.

4. The method of Claim 2. wherein operating parameters of said user agent are preserved for use at the new location proximate said mobile station.

5. The method of Claim 1, wher in said user process is anchored at said user signaling s rver.

6. The method of Claim 5, wherein said network includes one r more radio ports to enable wireless communication with said mobile station, further including the step of updating said radio ports with a location of said user agent as said mobile station associated with said user agent moves.

7. The method of Claim 5, wherein said network includes one or more radio ports to enable wireless communication with said mobile station, wherein signaling links are established between said radio ports and a user signaling server in which said user agent is resident as said mobile station moves throughout said network.

8. The method of Claim 3, wherein said network includes at least one location server for tracking the location of a mobile station within said network, said method further including the steps of:

generating a registration message from said mobile station to said location server upon changing clusters, wherein said location server signals to said target user signaling server to migrate said user agent of said mobile station; and
requesting and receiving user agent profile parameters at said target user signaling server from said first user signaling server.

9. The method of Claim 2, further including the step of maintaining a stub process at said first user signaling server to act in place of said user agent to buffer messages for said user agent during migration.

10. The method of Claim 9, wherein said stub process exits after a predetermined idle time-out period.

11. The method of Claim 9, wherein said user agent requests buffered information from said stub process once located at said second user signaling server, wherein normal processing by said agent is subsequently resumed.

12. The method of Claim 2, wherein said user agent is updated about a current state of said mobile station and said call processing functions include call acceptance and rejection and compatibility checking of data transmissions.

13. The method of Claim 2, wherein said user agent is a user process and said step of migrating further includes the steps of:

executing a new user process at said second user signaling server:
establishing a communications connection between a previous user process at said first user signaling server and said new user process;
transferring a data structure containing migration variables from said previous user process;
establishing communication channels at said new user process;
maintaining said previous user process as a stub process; and
forwarding messages received during migration to said new user process.

14. The method of Claim 13, wherein said data structure includes call and connection data and current state data elements of said mobile station, wherein said call connection data is selected from the group consisting of number of users, identification of users, call type, call forwarding and mobile call termination status, and said current state data elements of said mobile terminal include compatibility checking procedures and applications being run at said mobile station.

15. The method of Claim 2, wherein said user agent is a user process and said step of migrating further includes:

stopping and saving data register contents of a first user process on said first user signaling server at a source node;
transferring arguments and environment of said first user process to said server user signaling server;
creating a stub process to buffer incoming messages to said first user process;
executing a new user process on a target node at said second user signaling server with said arguments and environment obtained from said user signaling server of said source node;
calculating data and stack sizes of said first user process in migration at said first user signaling server of said sourc node;
transferring said data stack sizes to said target user signaling server to thereby enabl said new user process

to expand its stack and data pages;
transf rring and writing said data pages onto said new user process executing on said target user signaling s rver;
transferring and writing said stack pages;
establishing communication channels at said new user process;
sending a message from said new user process to said stub process and receiving outstanding messages that have been buffered; and
transferring said register contents, wherein said new user process restarts from where it was interrupted.

16. The method of Claim 2, wherein said user agent is a user process and said step of migrating further includes the steps of:

sending a stop signal from a target user signaling server to said first user process;
creating a child stub process that buffers messages sent to said first user process;
executing a save register command from a target user process to save registers and a core dump in a buffer;
transferring data and stack pages from said core to said target user signaling server;
creating a new user process at said target user signaling server using a text segment from an executable present at an associated node, and with stack and data segments received from said first user signaling server;
establishing communication channels at said new user process;
sending a message from said new user process to said stub process and receiving outstanding messages that have been buffered; and
executing a restore registers command at said new user process to enable continued execution from a point where said user process began migration.

17. A communications network including mobile stations within said network, wherein each of said mobile stations is associated with a cluster within said network, said network comprising:

at least one user signaling server associated with each said cluster of said network, said user signaling server coupled to a call server within said cluster, wherein said user signaling server includes,
at least one user agent, said user agent being associated with an active individual mobile station, said user agent operable to handle call processing functions on behalf of said mobile station thereby reducing air interface traffic between said mobile station and said user signaling server.

18. The network of Claim 17, wherein said user agent is further operable to migrate to another user signaling server in a different cluster when said mobile station moves to said different cluster.

19. The network of Claim 18, wherein each said cluster further includes:

at least one connection server for establishing a communication path in said cluster over which user voice and data information are transmitted;
at least one call server for maintaining the network state of a call within said cluster; and
at least one location server for tracking the location of a mobile station within said network, wherein said mobile station is operable to generate a registration message to an associated location server upon changing clusters, wherein said location server signals to a target user signaling server to migrate said user agent associated with said mobile station, and wherein said target user signaling server is operable to request and receive relevant user agent parameters from a previous user signaling server.

20. The network of Claim 18, wherein a previous user signaling server is operable to maintain a stub process in place of said user agent to buffer messages for said user agent during migration.

21. The network of Claim 20, wherein said stub process exits after a predetermined idle time-out period and wherein said user agent requests buffered information from said stub process once located at said target user signaling server, wherein normal processing by said agent is subsequently resumed.

22. The network of Claim 18, wherein said user agent operates a user process and said user process is operable to migrate by:

executing a new user process at a target user signaling server;

establishing a c mmunications connection between a previous user process at a first user signaling server and said new user process;
transferring a data structur containing migration variables from said previous user process;
establishing communication channels at said new user process;
maintaining said previous user process as a stub process; and
forwarding messages received during migration to said new user process.

FIG. 1

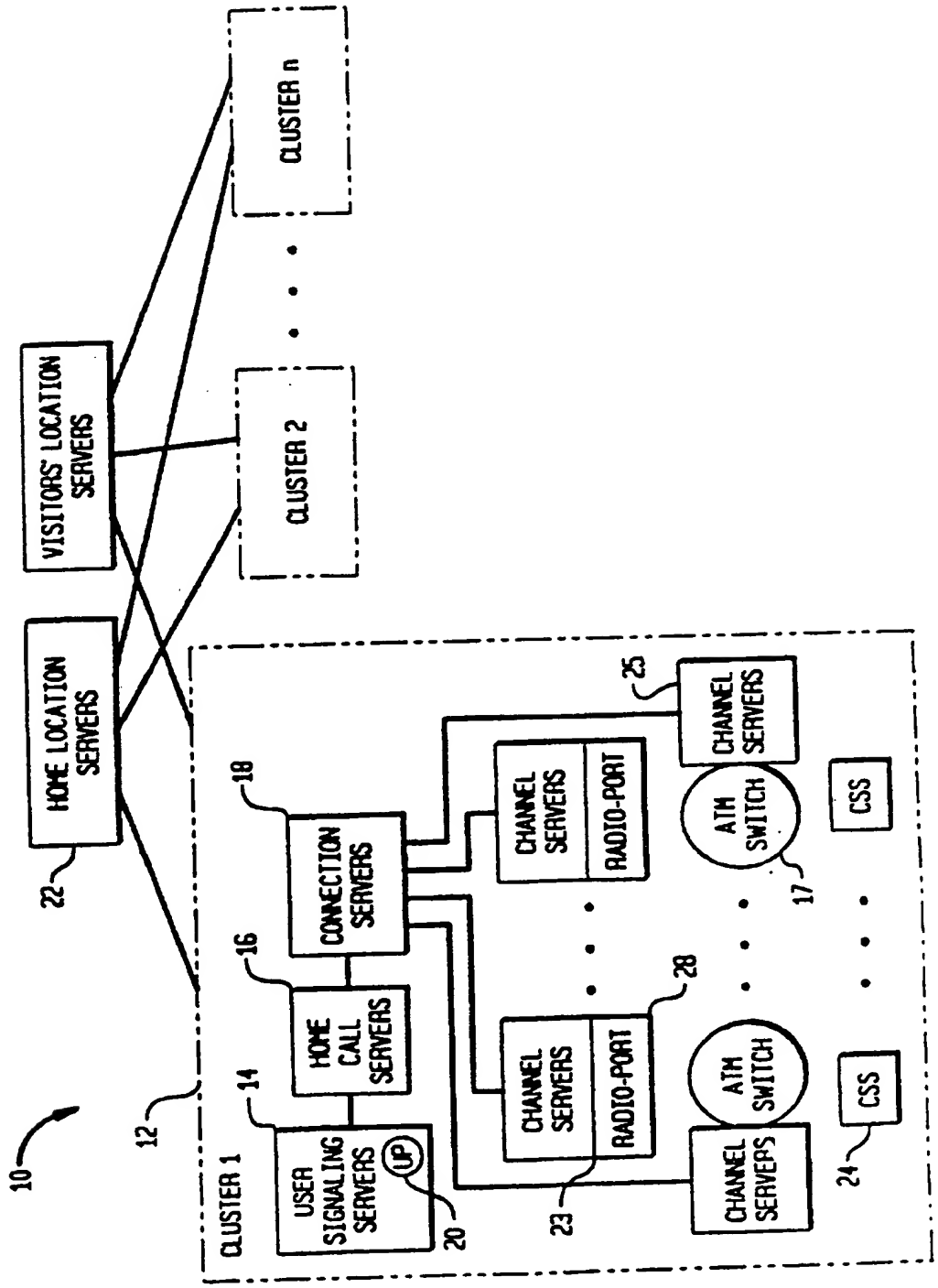

CLUSTER n
CLUSTER 2
VISITORS' LOCATION SERVERS
HOME LOCATION SERVERS
22
10
12
CLUSTER 1
14
USER SIGNALING SERVERS
UP
20
16
HOME CALL SERVERS
18
CONNECTION SERVERS
CHANNEL SERVERS
RADIO-PORT
23
28
CHANNEL SERVERS
25
ATM SWITCH
17
CSS
24

FIG. 2

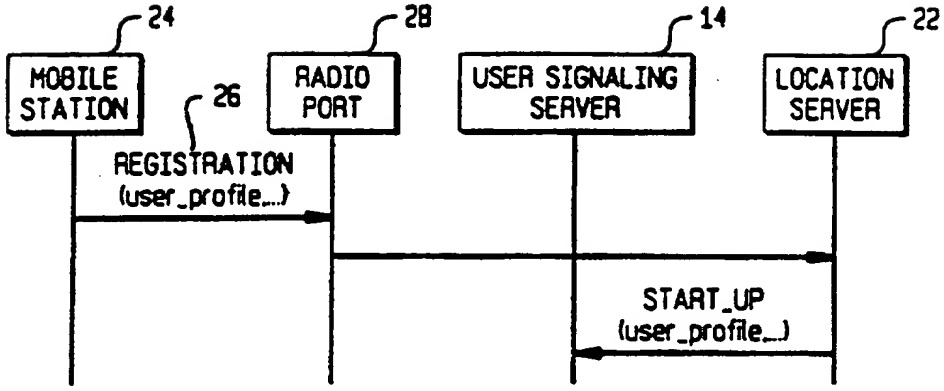

24
28
14
22
MOBILE STATION
26
RADIO PORT
USER SIGNALING SERVER
LOCATION SERVER
REGISTRATION (user_profile...)
START_UP (user_profile...)

FIG. 3

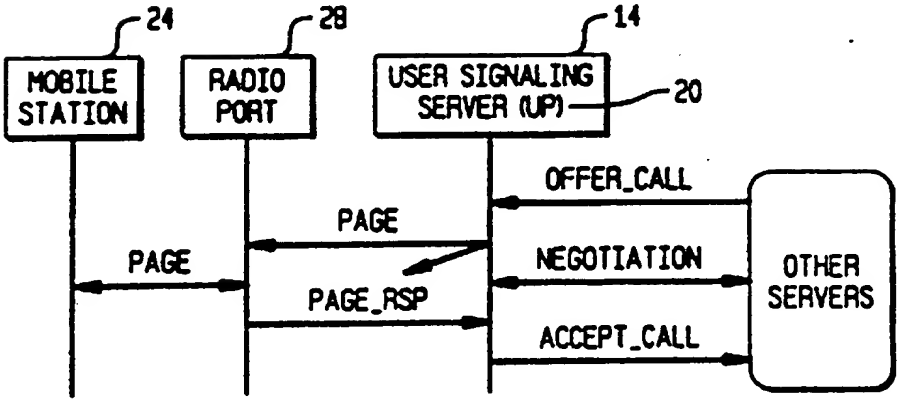

24
28
14
MOBILE STATION
RADIO PORT
USER SIGNALING SERVER (UP)
20
OFFER_CALL
PAGE
PAGE
NEGOTIATION
PAGE_RSP
ACCEPT_CALL
OTHER SERVERS

FIG. 4

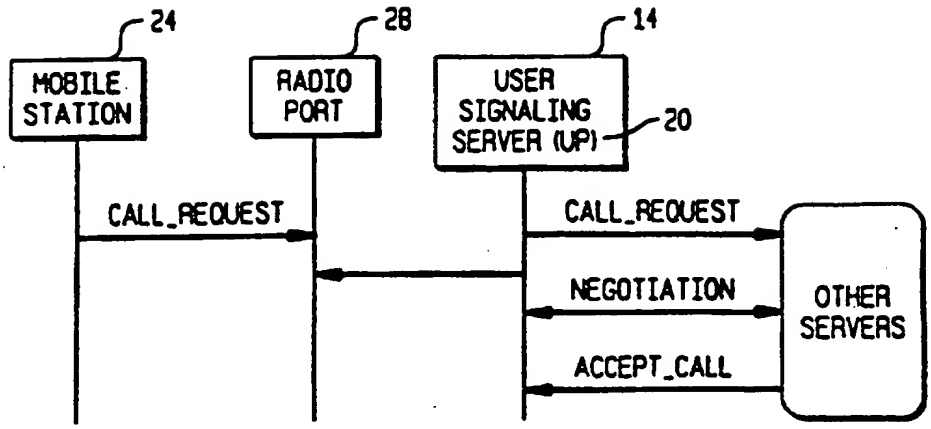

24
28
14
MOBILE STATION
RADIO PORT
USER SIGNALING SERVER (UP)
20
CALL_REQUEST
CALL_REQUEST
NEGOTIATION
ACCEPT_CALL
OTHER SERVERS

FIG. 5

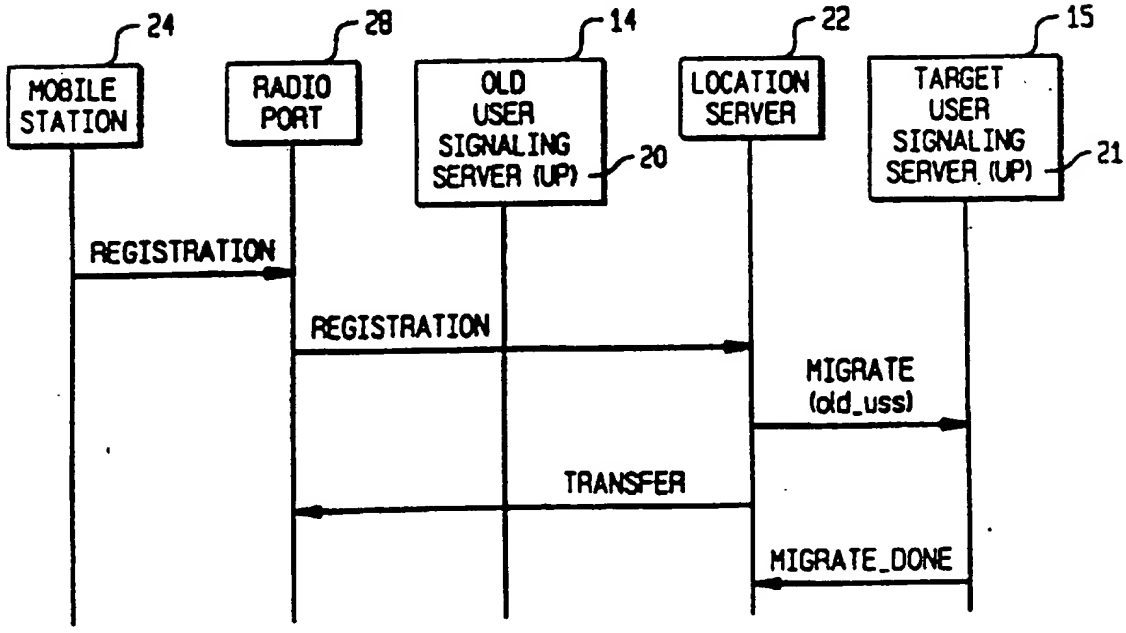

24
28
14
22
15
MOBILE STATION
RADIO PORT
OLD USER SIGNALING SERVER (UP)
20
LOCATION SERVER
TARGET USER SIGNALING SERVER (UP)
21
REGISTRATION
REGISTRATION
MIGRATE (old_uss)
TRANSFER
MIGRATE_DONE

FIG. 6

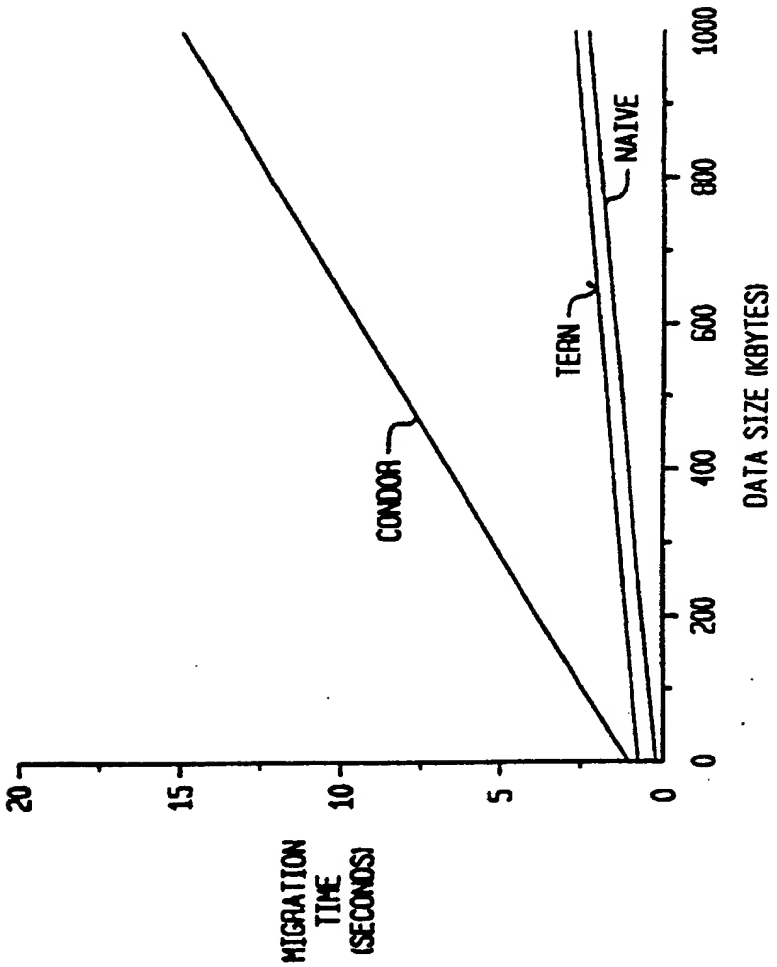

MIGRATION TIME (SECONDS)
20
15
10
5
0
CONDOR
TEAM
NAIVE
0
200
400
600
800
1000
DATA SIZE (KBYTES)

FIG. 7

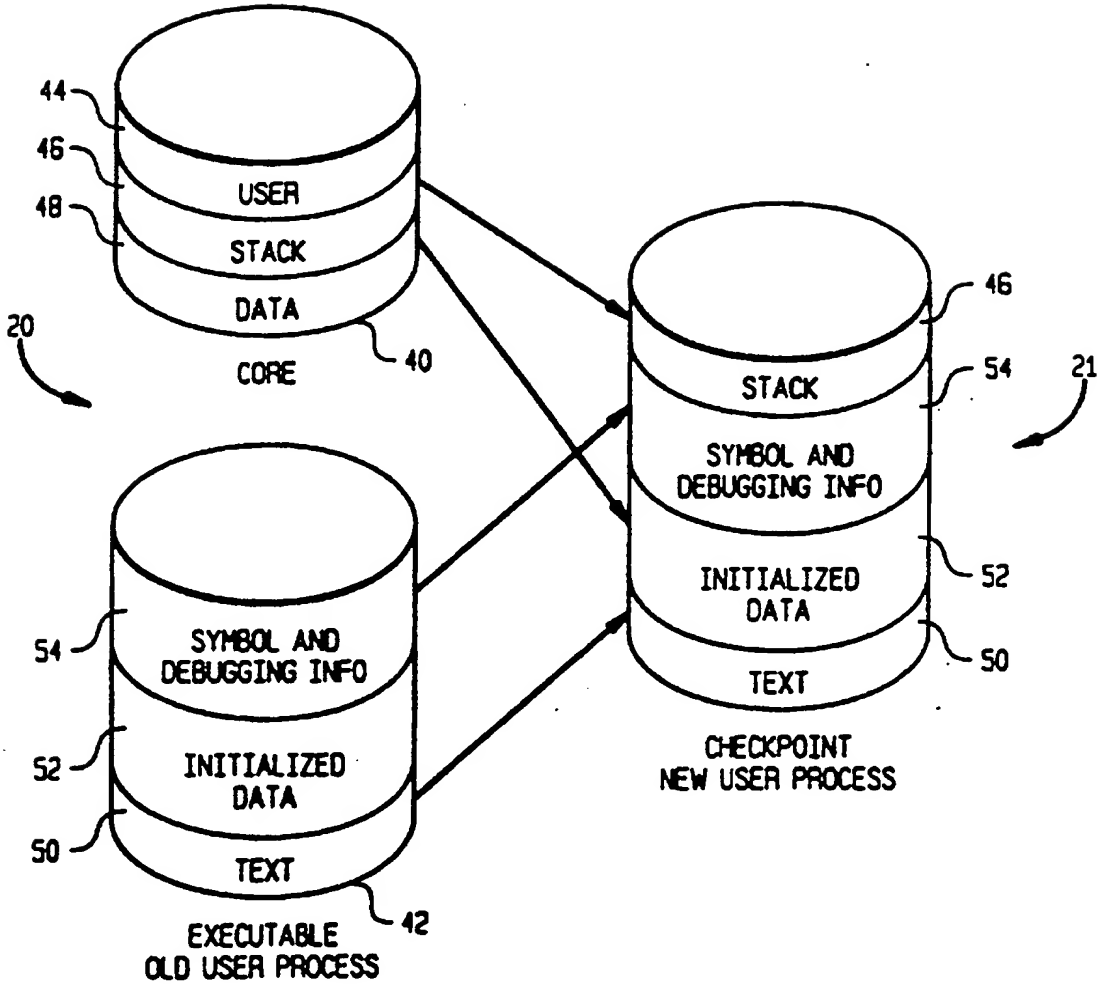

44
46
48
USER
STACK
DATA
40
CORE
20
54
SYMBOL AND DEBUGGING INFO
52
INITIALIZED DATA
50
TEXT
42
EXECUTABLE OLD USER PROCESS
46
54
21
STACK
SYMBOL AND DEBUGGING INFO
52
INITIALIZED DATA
50
TEXT
CHECKPOINT NEW USER PROCESS

FIG. 7A

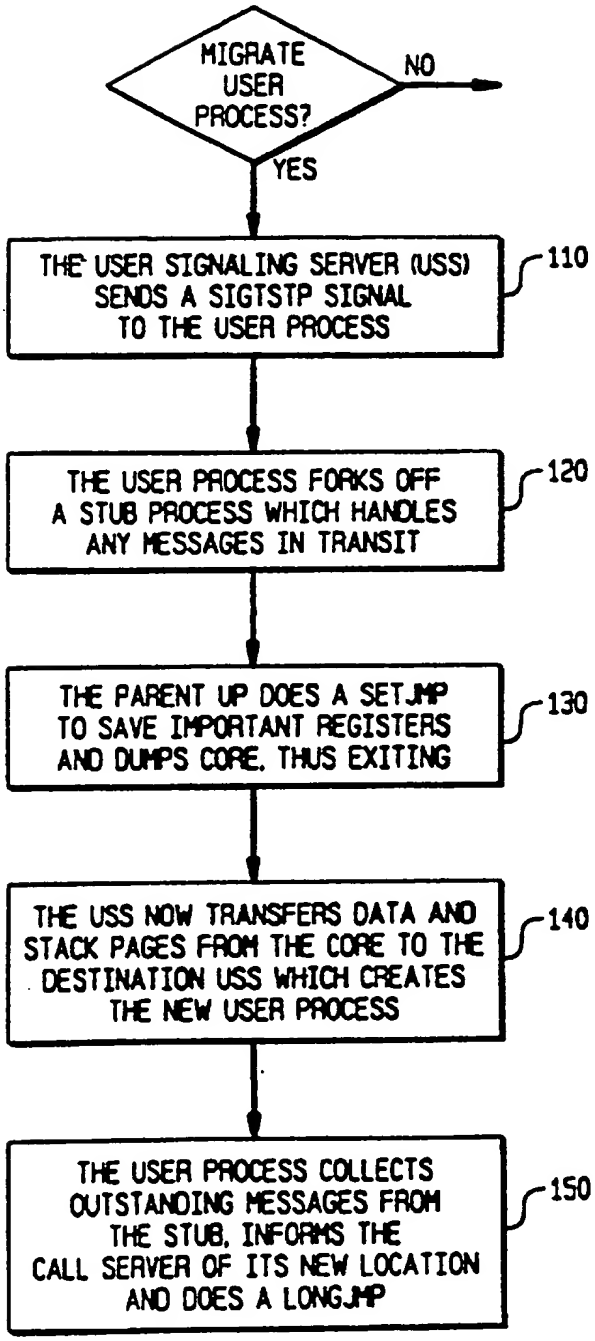

MIGRATE USER PROCESS?
NO
YES
THE USER SIGNALING SERVER (USS) SENDS A SIGTSTP SIGNAL TO THE USER PROCESS
110
THE USER PROCESS FORKS OFF A STUB PROCESS WHICH HANDLES ANY MESSAGES IN TRANSIT
120
THE PARENT UP DOES A SETJMP TO SAVE IMPORTANT REGISTERS AND DUMPS CORE. THUS EXITING
130
THE USS NOW TRANSFERS DATA AND STACK PAGES FROM THE CORE TO THE DESTINATION USS WHICH CREATES THE NEW USER PROCESS
140
THE USER PROCESS COLLECTS OUTSTANDING MESSAGES FROM THE STUB, INFORMS THE CALL SERVER OF ITS NEW LOCATION AND DOES A LONGJMP
150

FIG. 8

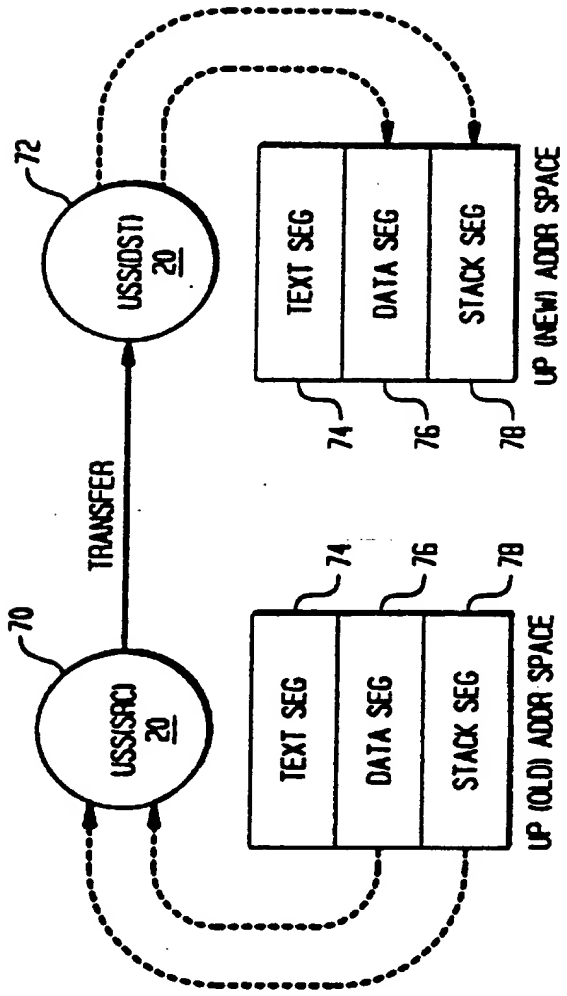

72
USSIDST)
20
TEXT SEG
DATA SEG
STACK SEG
UP (NEW) ADDR SPACE
74
76
78
TRANSFER
70
USSISRC)
20
TEXT SEG
DATA SEG
STACK SEG
UP (OLD) ADDR SPACE
74
76
78

FIG. 8A

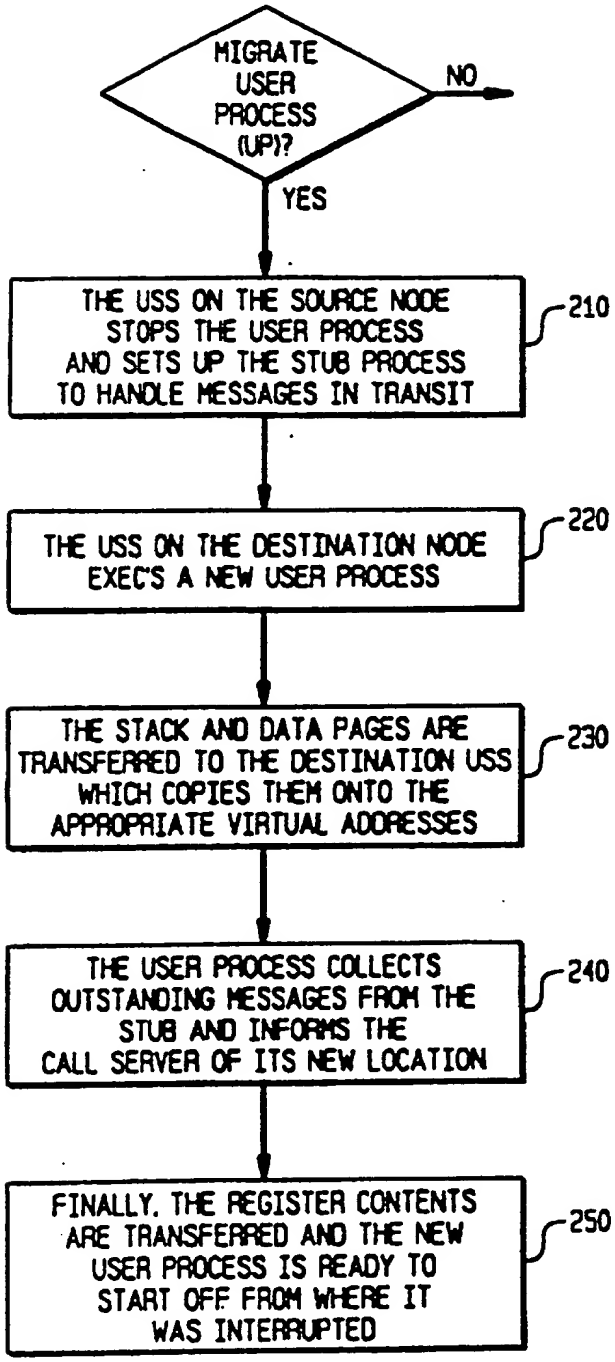

MIGRATE USER PROCESS (UP)?
NO
YES
THE USS ON THE SOURCE NODE STOPS THE USER PROCESS AND SETS UP THE STUB PROCESS TO HANDLE MESSAGES IN TRANSIT
210
THE USS ON THE DESTINATION NODE EXEC'S A NEW USER PROCESS
220
THE STACK AND DATA PAGES ARE TRANSFERRED TO THE DESTINATION USS WHICH COPIES THEM ONTO THE APPROPRIATE VIRTUAL ADDRESSES
230
THE USER PROCESS COLLECTS OUTSTANDING MESSAGES FROM THE STUB AND INFORMS THE CALL SERVER OF ITS NEW LOCATION
240
FINALLY, THE REGISTER CONTENTS ARE TRANSFERRED AND THE NEW USER PROCESS IS READY TO START OFF FROM WHERE IT WAS INTERRUPTED
250

FIG. 9

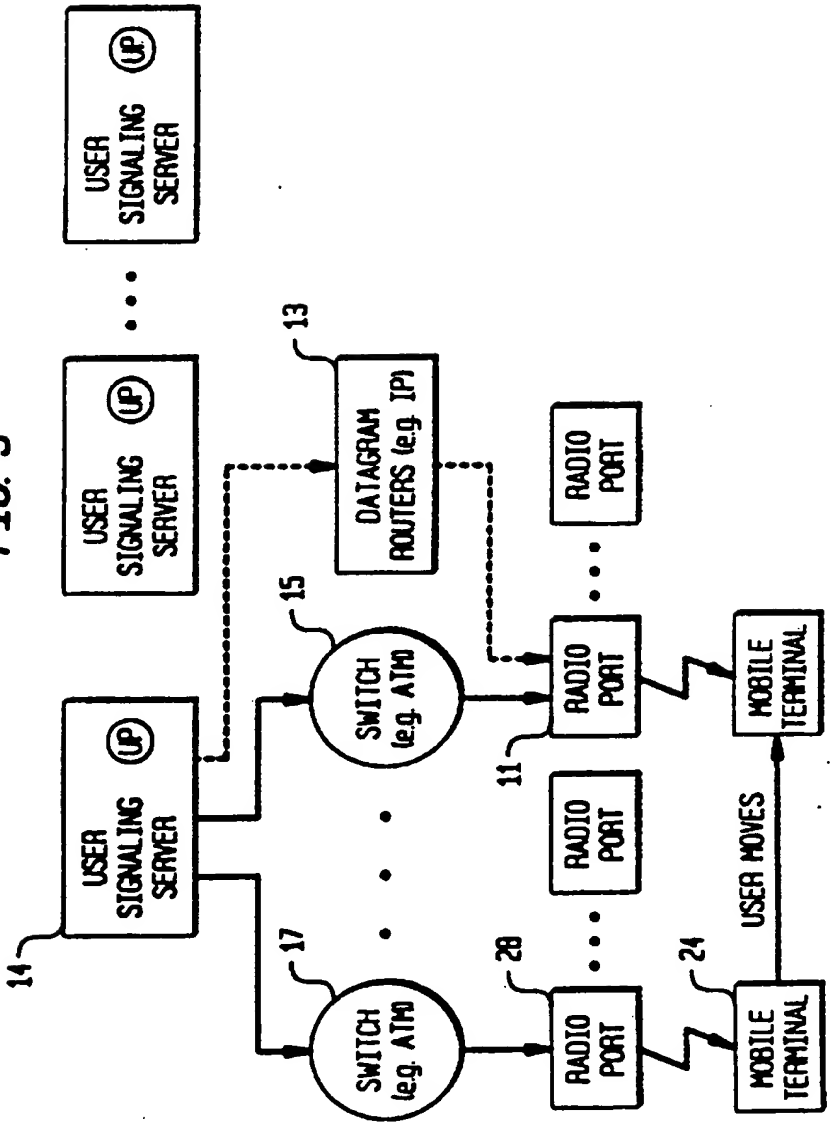
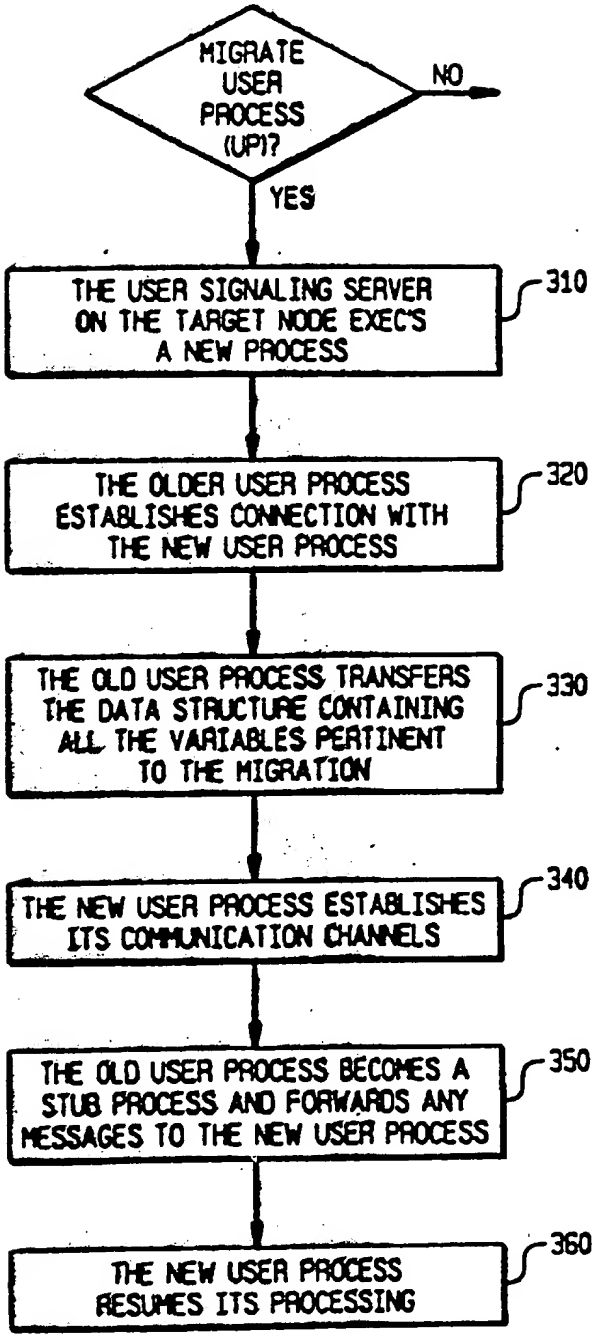

USER SIGNALING SERVER
UP
USER SIGNALING SERVER
UP
USER SIGNALING SERVER
UP
14
13
DATAGRAM ROUTERS (e.g. IP)
15
SWITCH (e.g. ATM)
17
SWITCH (e.g. ATM)
11
RADIO PORT
RADIO PORT
RADIO PORT
RADIO PORT
28
24
MOBILE TERMINAL
MOBILE TERMINAL
USER MOVES

FIG. 10


MIGRATE USER PROCESS (UP)?
NO
YES
THE USER SIGNALING SERVER ON THE TARGET NODE EXEC'S A NEW PROCESS
310
THE OLDER USER PROCESS ESTABLISHES CONNECTION WITH THE NEW USER PROCESS
320
THE OLD USER PROCESS TRANSFERS THE DATA STRUCTURE CONTAINING ALL THE VARIABLES PERTINENT TO THE MIGRATION
330
THE NEW USER PROCESS ESTABLISHES ITS COMMUNICATION CHANNELS
340
THE OLD USER PROCESS BECOMES A STUB PROCESS AND FORWARDS ANY MESSAGES TO THE NEW USER PROCESS
350
THE NEW USER PROCESS RESUMES ITS PROCESSING
360